

1                   A METHOD OF FAST FINGERPRINT SEARCH SPACE PARTITIONING AND  
2                   PRESCREENING

3                   BACKGROUND OF THE INVENTION

4       1. Field of the Invention

5       This invention generally relates to fingerprint matching systems in which a fingerprint is matched  
6       to reference fingerprints in a database, and more particularly, to a fingerprint matching system  
7       for rapidly locating matching fingerprints in a repository of fingerprint data containing a million or  
8       more fingerprints by pre-screening the repository of fingerprint data for likely matching  
9       fingerprints to create an index or list of candidate mated fingerprints.

10      2. Background of the Invention

11      Pattern matching or comparison schemes have many applications such as the matching of  
12      fingerprints for comparison with file fingerprints. Fingerprints are very rich in information content  
13      and basically contain two major types of information: 1) the ridge flow information, and 2) the  
14      specific features or minutiae (minutia) of the fingerprint. As used herein , the fingerprint to be  
15      identified may be termed an "unknown" fingerprint or a "latent" fingerprint.

16      Fingerprints uniquely identify an individual based on their information content. Information is  
17      represented in a fingerprint by the minutia and their relative topological relationships. The  
18      number of minutia in a fingerprint varies from one finger to another, but, on average, there are  
19      about eighty (80) to one hundred and fifty (150) minutia per fingerprint. In the fingerprint context,  
20      a large store of fingerprints exists in law enforcement offices around the country. These  
21      fingerprints include files of fingerprints of known individuals, made in conjunction with their  
22      apprehension or for some other reason such as security clearance investigation or of obtaining  
23      immigration papers, often by rolling the inked fingers on cards, and also includes copies of  
24      latent fingerprints extracted from crime scenes by various methods.

25      These reference fingerprints are subject to imperfections such as overinking, which tends to fill  
26      in valleys in fingerprints, and underinking, which tends to create false ridge endings, and  
27      possibly both overinking and underinking in different regions of the same fingerprint image.  
28      Smudging and smears occur at different places in the fingerprint due to unwanted movement of  
29      the finger, or uneven pressure placed on the finger, during the rolling process. The stored  
30      fingerprints are also subject to deterioration while in storage, which may occur, for instance, due

Mailing Label # ET129576494US

1 to fading of the older images, or due to stains. Furthermore, the wide variation in the level of  
2 experience among fingerprint operators, and the conditions under which the fingerprint is  
3 obtained, produces wide variation in quality in the fingerprint images. Similar effects occur due  
4 to the variation of the scanning devices in cases of live scanning of fingerprints.

5 Matching of fingerprints in most existing systems relies for the most part on comparison of cores  
6 and deltas as global registration points, which tends to make the comparisons susceptible to  
7 errors due to the many sources of distortion and variations listed above, which almost always  
8 occur due to the various different inking, storage and reprocessing conditions which may be  
9 encountered.

10 As described at pages 164-191 of the text Advances in Fingerprint Technology, by Henry C. Lee  
11 and R. E. Guenssten, published by Elsevier in 1991, efforts have been underway for a long time  
12 to automate fingerprint identification, because manual search is no longer feasible due to the  
13 large number of reference files. The effort to automate fingerprint identification involves two  
14 distinct areas, namely (a) that of fingerprint scanning and minutia identification, and (b)  
15 comparison of lists of minutia relating to different fingerprints in order to identify those which  
16 match. Large files of reference fingerprints have been scanned, and minutia lists in digital form  
17 obtained therefrom, either by wholly automated equipment, or with semi-automated equipment  
18 requiring human aid. While not all problems in scanning of fingerprints and detection of minutia  
19 have been solved, it appears that the matching problem is the more pressing at this time.

20 The matching or search subsystem constitutes the most critical component of any Automated  
21 Fingerprint Identification System (AFIS). Its performance establishes the overall system  
22 matching reliability (the probability of declaring the correct mate, if one exists in the database),  
23 match selectivity (the average number of false candidates declared in each search attempt),  
24 and throughput, which is particularly important in large database systems. The unique  
25 identification of fingerprints is usually performed using the set of minutia contained in each  
26 fingerprint.

27 U.S. Pat. No. 5,613,014, issued Mar. 18, 1997 in the name of Eshera et al. describes a  
28 fingerprint matching technique using a graphical attribute relational graph (ARG) approach. This  
29 ARG approach is fast, and particularly advantageous for those cases in which the minutia of the  
30 latent or unknown fingerprint are numerous and well defined, but may be hindered in finding the

1 correct match by errors in locating minutia near the center of each star when the latent image is  
2 poor and minutia are missing.

3 However, because the fingertip skin is flexible, the relative locations and orientation of the  
4 pattern singularities and minutiae differ (at least slightly) from one impression of a given finger to  
5 the next under controlled conditions (for example from multiple rolled prints of the same finger).  
6 These differences are magnified in latent, unknown fingerprints which are not made with the  
7 assistance of a fingerprint operator, but rather may be left a crime scene on different types of  
8 surfaces, such as flexible surfaces, under vastly differing pressures, with some times only a  
9 fraction of the finger area being involved. This invention addresses the problem of identifying  
10 candidate mate fingerprints in a repository for either rolled or latent search fingerprints.

### 11 SUMMARY OF THE INVENTION

12 Accordingly, it is an object of the present invention to overcome the deficiencies of the prior art  
13 in addressing the problem of identifying candidate mate fingerprints in a repository for either  
14 rolled or latent search fingerprints.

15 Yet another object of the present invention is to provide a method for fast fingerprint  
16 identification which rapidly locates matching fingerprints in a repository of fingerprint data  
17 containing a million or more fingerprints.

18 Yet another object of the present invention is to provide a method of fingerprint identification  
19 whereby a large repository of fingerprints is very rapidly searched for members of the repository  
20 that most nearly match the search print to create a list of candidate mate fingerprints that are  
21 then more carefully search for matching features.

22 Still another object of the present invention is to provide a method of fast fingerprint  
23 identification wherein the index is based on each minutia and selected neighbors of each  
24 minutia in each file fingerprint in the repository and then the index is subsequently searched to  
25 identify all minutiae which correspond to the minutiae in a search fingerprint. The results of this  
26 search are analyzed to determine which file fingerprints contributed the most minutiae with the  
27 best correspondence to the minutiae in the search fingerprint.

28 These and other objects, advantages and features of the present invention are achieved by a  
29 method comprising the steps of: inputting the contents of a fingerprint repository comprising file  
30 fingerprints for index creation; creating an index based on each minutia and selected neighbors

of each minutia in each file fingerprint in the repository; searching the index to identify all minutiae which correspond to the minutiae in a search fingerprint; and analyzing results of this search to determine which file fingerprints contributed the most minutiae with the best correspondence to the minutiae in the search fingerprint.

#### **BRIEF DESCRIPTION OF THE DRAWINGS**

Figure 1 is a block diagram representing repository index data;

Figure 2 is a block diagram illustrating the steps for generating an index of fingerprint files from the repository;

Figure 3 is a block diagram illustrating the steps of the CreateIndexfiles subroutine for creating each of the files of the a specific index;

Figure 4 illustrates the data contained in a hash list node;

Figures 5 and 6 illustrate the steps of the ExactMatch subprogram;

Figure 7 illustrate the steps of the AddSubject subprogram;

Figure 8 illustrates the steps of the GenerateHashCode subprogram;

Figure 9 illustrates a typical quantization vector;

Figure 10 illustrates a typical equalization vector;

Figure 11 illustrates a typical equalization matrix;

Figure 12 illustrates the steps of the AddSubjectToList program;

Figures 13 illustrate the steps of the SearchIndex program;

Figures 14-15 illustrate the steps of the IndexSearch subprogram;

Figure 16 illustrates the steps of the VisitMatch subprogram;

Figure 17 illustrates transfer vector data;

Figure 18 illustrates match data;

1 Figure 19 illustrates the steps of the AccumulateHough subprogram; and

2 Figures 20 and 21 illustrate the steps of the EvaluateMatch subprogram.

3 **DETAILED DESCRIPTION OF THE PREFERED EMBODIMENT(S)**

4 The following terms used in this disclosure are defined as set forth below.

5 Fingerprint characteristics - Data describing a fingerprint that has been automatically or  
6 manually extracted from an image of the fingerprint. This data includes but is not limited to the  
7 Pattern Classification and a set of Fingerprint Minutiae (typically 10 to 200).

8 Fingerprint Repository - One or more files containing the characteristics of multiple fingerprints.

9 Fingerprint Minutiae - Endings or Bifurcations in the friction ridges of a fingerprint. Also known  
10 as Galton points or Level 2 details.

11 Pattern Classification - Enumeration of the general patterns of the flow of the friction ridges of a  
12 fingerprint, the most common being arches, loops and whorls. Also known as Level 1 details.

13 Minutia Data - Data describing the position, type, and orientation of a given minutia, and the  
14 relationship of that minutia to its neighboring minutiae.

15 Pattern Singularities - Discontinuities in the general flow of the friction ridges of a fingerprint, the  
16 most common being cores and deltas. The number, types and relative locations of the Pattern  
17 Singularities determine the Pattern Classification.

18 File Print - A fingerprint whose characteristics have been stored in a repository.

19 Search Print - A fingerprint that is being sought in the repository.

20 Mate (Print) - The File Print (or prints) which correspond to the subject (person) who generated  
21 the Search Print.

22 Rolled Print - A fingerprint obtained by rolling the subject finger across the acquisition surface.

23 File prints are almost exclusively rolled prints. Search prints may or may not be rolled prints.

24 Latent Print - A fingerprint impression left at the scene of a crime or developed into an image by  
25 investigators.

1 In accordance with the teachings of the present invention, there is provided a method for rapidly  
2 locating matching fingerprints in a repository of fingerprint data containing a large number of  
3 fingerprints wherein the method creates an index into the repository which is based on each  
4 minutia and selected neighbors of each minutia in each file fingerprint in the repository. The  
5 index is subsequently searched to identify all minutiae that correspond to the minutiae in a  
6 search fingerprint. The results of this search are analyzed to determine which file fingerprints  
7 contributed the most minutiae with the best correspondence to the minutiae in the search  
8 fingerprint.

9 The input data for the creation of the index are the contents of a fingerprint repository (or any  
10 fingerprints that are to be added to an existing repository), or a set of characterized search  
11 fingerprints. For each fingerprint (either file or search) processed according to one embodiment  
12 of the method of the present invention, the following data are used:

13 ❖ The number of minutiae in the fingerprint.

14 ❖ For each minutia:

- 15     ▪ The base angle of the minutia in a frame of reference whose angular orientation to  
16     and permissible deviation from the axis of the finger is pre-defined by customer  
17     specification.
- 18     ▪ The Cartesian coordinates (X, Y) of the minutia relative to the same frame of  
19     reference.
- 20     ▪ The minutia index of the nearest neighboring minutia (if any) in each of eight octants  
21     (45 degree wedges), the first octant of which is centered on the base angle of the  
22     minutia. (These minutiae are known as "octant neighbors"). Note that minutiae  
23     which fall near the edge of the fingerprint may not have neighbors in all octants.
- 24     ▪ The count of friction ridges between the minutia and each of its eight octant  
25     neighbors.
- 26     ▪ The Euclidean distance between the minutia and each of its eight octant neighbors.
- 27     ▪ The difference between the base angle of the minutia and the base angle of each of  
28     its eight octant neighbors.

1 It should be noted that the above example describes the implementation based on ARG data  
2 used in the patented LMIS ARG Matcher as disclosed by U.S. Patent No. 5,613,014, issued  
3 March 18, 1997, the entire disclosure of which is herein incorporated by reference. Other  
4 implementations, which are envisioned to be within the scope of the present invention, may use  
5 additional (or different) data and still work as well or better than the currently described  
6 embodiment. For example, high-resolution direction-to-the-octant-neighbor can be included to  
7 improve the speed, reliability and selectivity of the algorithm, providing those data are available  
8 in the repository. However, it should be noted that a primary feature of the method of the  
9 present invention is that data for neighboring minutiae are used in the index.

10 Each minutia in the repository, and in any fingerprint that is added to the repository, is uniquely  
11 identified by a repository index, shown in Figure 1, which contains the following information:

- 12 • The index of the subject in the repository (typically starting at zero and  
13 increasing by one as each subject is added to the repository).
- 14 • The index of the finger within the subject (typically starting at zero for the right  
15 thumb and continuing through 9 for the left little finger).
- 16 • The index of the minutia within the subject and finger (typically starting at  
17 zero and increasing by one until the maximum number of minutiae allowed in  
18 the repository is reached).

19 In accordance with the teachings of the present invention, the method creates the data files  
20 associated with the index and adds from one to all of the fingerprints in the repository to the  
21 index. Referring to Figures 2, in step 1, the repository file(s) in the repository R are opened for  
22 read-only access. In step 2, a determination is made as to whether or not index files exist. In  
23 step 2A, if the index files do not already exist, the subprogram CreateIndexFiles, as will be more  
24 fully described with respect to Figure 3, creates the index files and opens them for read-write  
25 access. Otherwise, in step 2b the files of the existing index are opened for read-write access. A  
26 user-specified set of repository subjects can also be added manually to the index, in accordance  
27 with known methods, for example, without limitation, using records in control files, or fields in a  
28 graphic user interface or the like.

29 In step 3, for each subject S in the repository of fingerprint files to be added to the index,  
30 Subprogram ExactMatch is used to determine whether subject S is already in the index. This is

done to avoid introducing duplicate records in the index, since the presence of duplicate records would bias the search results towards the selection of those subjects duplicated in the index.

In step 3a, if the subject S is not already in the index, subprogram AddSubject is used to add subject S to the index. In step 4, once all subjects S in the repository have been scanned and the non-duplicative subjects have been added to the index, the repository and index files are closed.

Figure 3 illustrates the individual steps of the CreateIndexFiles subprogram used to create the all the files needed for a specific index I. In this regard, it is important to note that an index can be created for any subset of ten fingers. Typically, the two index fingers are used for rolled search capability, however, all ten fingers is needed for latent search capability where the source finger number is unknown. In step Step 2(a)1 of the CreateIndexFiles subroutine, index neighborhood combinations are created for each finger of the index. Neighbor combinations are based on selecting 2 of eight octant neighbors to produce 28 possible combinations per finger, as shown in Table1. In one embodiment of this subprogram of the present invention, all combinations of two neighbors (N=2) out of 8 neighbors are used, however N may be 1, 2, 3, 4, 5, 6 or 7. For the available LMIS ARG data, N=2 is the best, but N=3 has potential value for other fingerprint data, for example, when there is no ridge count data, in which case N=3 is better than N=2.

Combination	Minutia A	Minutia B
0	0	1
1	0	2
2	0	3
3	0	4
4	0	5
5	0	6
6	0	7
7	1	2
8	1	3
9	1	4
10	1	5
11	1	6
12	1	7



13	2	3
*	*	*
24	5	6
26	5	7
27	6	7

Table 1. All combinations of two neighbors out of eight neighbors.

In step 2(a)2, one Hash Table file T is created for each Finger/Neighbor Combination. A hash table consists of 32,768 entries, each of which contains the following data:

- ❖ The index of a hash list node in the hash list file.
- ❖ The number of hash list nodes associated with the hash code

There are 56 Hash Table files T[2][28]. In this regard, the 32,768 entries are based on a 15-bit hash code. This is probably the smallest practical hash code for fingerprint applications and is based on the LMIS ARG data. If other data were available, the number could increase to more than 4 million, which would give better performance. The data elements of an entry are independent of the number of entries (they would still be a hash list node index and a hash list node count).

In step 2(a)3, one Hash List file L is created for each Finger/Neighbor Combination. A hash list consists of N hash list nodes, where N is a function of the number of minutiae in the repository, plus any room needed for repository expansion. Each Hash List Node contains the following data, as shown also in Figure 4:

- ❖ The Hash Code associated with the buffer (needed for data reliability checking, but otherwise not used in the algorithm).
- ❖ The number of populated repository index (Figure 1) slots in the buffer.
- ❖ Thirty repository index slots.

There are 56 Hash List files L[2][28]. The optimum number of slots in a node is a function of the hash code distribution that, in turn, is a function of the fingerprint characteristics in the repository. For the LMIS ARG data the absolute best is 24 slots, but 30 is nearly as good and

1 offers the additional benefit of producing a node size that is a power of 2 and thus much more  
2 efficient in terms of practical disk operations.

3 In step 2(a)4, one Hash List Node Used file U is created for each Finger/Neighbor Combination.  
4 A Hash List Node Used file is a bookkeeping device which tracks which hash list nodes in the  
5 file are populated at any given time. There is exactly one bit in the Hash List Node Used file for  
6 each node in the associated Hash List file. The bit associated with a given node is set to 1  
7 when that node is in use and cleared to 0 when the node is available for use. There are 56 Hash  
8 List Node Used files U[2][28].

9 One embodiment of the details of step 3 shown in Figure 2 relating to the ExactMatch  
10 subprogram are shown in Figures 5 and 6. As previously noted, the ExactMatch subprogram  
11 determines if a specific subject S from the Repository R is already present in the Index I. This  
12 subprogram takes advantage of the fact that a duplicate subject in the repository is, by  
13 definition, guaranteed to have the identical hash code and the identical repository index as one  
14 already in the index. Therefore, the presence of the first existing combination associated with  
15 the first neighbor of the first minutiae of the first finger of the subject is indicative of a duplicate  
16 while the absence is indicative that the subject is not in the index.

17 The first set of process steps in the embodiment of the ExactMatch subprogram shown in Figure  
18 5 are used to identify the first existing combination associated with the first neighbor of the first  
19 minutiae of the first finger of the subject. The hash code for that neighbor combination is  
20 calculated and used to search the appropriate hash table. If there are no hash list nodes  
21 associated with the hash code, there is no duplicate entry in the index, ExactMatch is false and  
22 the subprogram terminates.

23 If there are hash list nodes associated with the hash code, the subprogram reads them from the  
24 disk and searches each in order until either the subject index is found or the end of the list is  
25 reached. If the subject index is found, ExactMatch is true else ExactMatch is false; in both  
26 cases, the subprogram terminates.

27 A detail description of one embodiment of the individual process steps of the AddSubject  
28 subprogram of Step 3a of the method of the present invention are shown in Figure 7. As  
29 previously noted with particular reference to Figure 2, the AddSubject subprogram adds each  
30 existing neighbor combination of each minutia of each finger of the subject S into the index I.

1 As seen in Figure 7, the AddSubject subprogram first selects the Finger F, Neighbor  
2 Combination C and Minutia M of each of the subject(s) being added (S). Note that F and C are  
3 used to select the appropriate Hash Table  $T[F][C]$ , Hash List  $L[F][C]$  and Hash List Node Used  
4  $U[F][C]$  files. If the Neighbor Combination C exists for Minutia M, the following steps are taken:  
5 1) Generate a hash code from the minutia and neighbor parameters and locate the hash code  
6 entry in the Hash Table  $T[F][C]$ . 2) If no prior hash list nodes are associated with the hash code,  
7 identify the next available node in the Hash List file  $L[F][C]$ , store its index in  $T[F][C]$  and update  
8  $U[F][C]$  appropriately. If prior hash list nodes are associated with the hash code, read them into  
9 memory and 3) Invoke AddSubjectToList to insert the subject (S) repository index into the first  
10 available slot in the Hash List  $L[F][C]$ .

11 The steps of one embodiment of the GenerateHashCode subprogram are shown by the flow  
12 diagram of Figure 8. The GenerateHashCode subprogram produces a 15-bit hash code from  
13 the parameters of a specified Minutia M and neighbor combination C. The neighbors in the flow  
14 diagram and this section are labeled Neighbor A and Neighbor B. The values of A and B are  
15 selected, as a function of neighbor combination C, from Table 1. The ideal hash code  
16 generation would cause each of the 32,768 possible values to be equally likely. The algorithm  
17 used here approaches, but does not meet this ideal, due primarily to the distribution of the  
18 fingerprint characteristics in the repository.

19 The Base Angle, Euclidean Distances to neighbors A and B and the Relative Angles to  
20 neighbors A and B of minutia M are quantized using Quantization Vectors as shown in Figure 9.  
21 The contents of the Quantization Vectors are selected based on the distribution of parameters in  
22 the repository R. Experimental evidence shows that they may be generated from a subset of R  
23 and left unchanged as R grows.

24 It should be noted that the quantization vectors are created by analyzing, on a minutia-by-  
25 minutia basis, the difference between mated pairs of fingerprint rollings. This is done by  
26 calculating the mean and standard deviation of the difference between each parameter. The  
27 value of the mean sets the starting point of the vector. The standard deviation is used to select  
28 the quantization interval. The value of the quantization vector components could change slightly  
29 based on the characteristics of a particular set of fingerprints (i.e., if one wanted to optimize the  
30 performance on fingerprints of males vs. females) or, in the case of the base angle, on the  
31 definition of "oriented" fingerprints as being +/- some angle.

The Ridge Counts to neighbors A and B, and the Euclidean Distances to neighbors A and B are histogram equalized using Equalization Vectors, as shown in Figure 10. As with the contents of the Quantization Vectors, the contents of the Equalization Vectors are selected based on the distribution of parameters in the repository R. Experimental evidence shows that they may be generated from a subset of R and left unchanged as R grows. The Equalization Vectors and Matrices are important only in the sense that they reduce the size of the hash code (i.e., 15 bits compared to say 20 bits) without significant impact on performance. The reduction of the size of the hash code reduces the amount of memory needed. Were memory not an issue, the Equalization Vectors can be eliminated and the larger hash code/memory usage accepted.

The Relative Angle to neighbors A and B are histogram equalized into a single value using an Equalization Matrix as shown in Figure 11. Again, the contents of the Equalization Matrix are selected based on the distribution of parameters in the repository R. Experimental evidence shows that they may be generated from a subset of R and left unchanged as R grows. Once all of the parameters have been quantized and equalized, they are combined into a single hash code as shown in Equation 1.

$$C_h = A_0 + R_0 (A_1 + R_1 (A_2 + R_2 (A_3 + R_3 (A_4 + R_4 (A_5))))))$$

$C_h$  is the generated Hash Code

$A_0$  is the quantized Base Minutia Direction

$R_0$  is the range of the Base Minutia Direction

$A_1$  is the equalized Neighbor A Ridge Count

$R_1$  is the range of the equalized Neighbor A Ridge Count

$A_2$  is the range of the equalized Neighbor B Ridge Count

$A_3$  is the equalized, normalized Neighbor A Euclidean Distance

$R_3$  is the range of the equalized normalized Neighbor A Euclidean Distance

$A_4$  is the equalized, normalized Neighbor A Euclidean Distance

$A_5$  is the two-dimension equalized, normalized Neighbor A and Neighbor b relative Directions

Equation 1. Hash Code Generation.

The AddSubjectToList subprogram flow diagram is shown in Figure 12. The AddSubjectToList subprogram presumes that the hash list node data for a given hash code is resident in memory.

1 Prior to invocation, existing nodes for a given hash code must be read into memory. Prior to  
2 invocation, if there are no pre-existing nodes, the memory buffer for the first node must be  
3 cleared. Each node in the list is searched for an available slot. If an available slot is found, the  
4 subject S's repository index is inserted in the slot, the node containing the slot is written to the  
5 file and done is set true.

6 If done is false, the implication is that all pre-existing nodes are full. In this case, a new node is  
7 appended in memory, and the subject S's repository index is inserted in the first slot. Since a  
8 node has been added, the list will no longer fit in its original location. The Hash List Node Used  
9 file U[F][C] is searched for the first available set of contiguous nodes which fit the new list size.  
10 The list data are written into these nodes. The hash table T[F][C] is updated, the previous node  
11 list locations are cleared and U[F][C] is updated to reflect the new node usage.

12 Once an index is created, step 5 of the method of the present invention is to locate the most  
13 likely candidate mate(s) for search subject S in index I. Step 5 of the method of the present  
14 invention comprises, for example, the SearchIndex program, a flow diagram of which is shown  
15 in Figure 13. The SearchIndex program searches for the mates, in the index I, of each entry in a  
16 list of search subjects. This program includes "truth" data to allow the performance of the  
17 IndexSearch subprogram, which implements the search process itself, to be evaluated. A list of  
18 search subjects is read, the index is searched for each subject, the performance is evaluated  
19 and a report generated.

20 The IndexSearch subprogram flow diagram is shown in Figures 14 and 15. The IndexSearch  
21 subprogram reads a search feature vector V, identifies all repository subjects that contain  
22 similar minutiae (candidate mates), then evaluates the candidate mates, selecting the best  
23 possible matches (if any) and appending them to the result list. After reading the search feature  
24 vector V, the subprogram searches the appropriate finger(s) F, minutiae M and neighbor  
25 combinations C. If the neighbor combination C exists for minutia M of finger F, a hash code is  
26 generated for the minutia and neighbor parameters. If the hash code entry in Hash Table  
27 T[F][C] contains list nodes, the nodes are read from the Hash List file L[F][C]. The VisitMatch  
28 subprogram is invoked for every repository index in every list node for the hash code.  
29 VisitMatch (described later) accumulates match data for every candidate mate in the search.

30 When all of the minutiae and neighbor combinations have been processed, in step 6 of the  
31 method of the present invention, EvaluateMatch is invoked to score and rank all of the

1 candidate mates. EvaluateMatch (described later) generates the list of the most likely mates  
2 from all of the candidates.

3 The VisitMatch subprogram flow diagram is shown in Figure 16. The VisitMatch subprogram is  
4 invoked for all candidate mates in the hash list. It is called with the repository index which  
5 contains the subject index S, the finger index F and the minutia index M for each candidate  
6 mate. It maintains a transfer vector, whose index is the subject index field of the repository  
7 index, that points to a candidate mate evaluation structure as shown in Figure 17. The purpose  
8 of the transfer vector is to minimize the amount of memory that must be cleared at the  
9 completion of each search. The entire transfer vector must be cleared, but only those  
10 MatchData entries that were used in the search need be cleared. Since the MatchData entries  
11 are significantly larger than the TransferVector entries, there is a net reduction in memory  
12 addresses which are cleared.

13 Each MatchData entry consists of the following data for each finger used for the search (also  
14 shown in Figure 18):

- 15 ❖ The VisitCounter which is incremented every time a given Subject/Finger is visited.
- 16 ❖ The MinutiaBitMap which contains one bit for each expected minutia; the appropriate bit is  
17 set when a given minutia participates in the match. The number of bits set matches the  
18 number of individual minutiae that participated in the match operation.
- 19 ❖ The Hough Accumulator (HoughAcc) which is used to accumulate the search minutiae-file  
20 minutiae relationships via a multi-dimensional Hough Transform.
- 21 ❖ The MatchScore which is used during match evaluation to provide a single number whose  
22 value is proportional to the degree of match between the search fingerprint and the file  
23 fingerprint.

24 The VisitMatch subprogram checks the TransferVector for search subject S. If the value is zero,  
25 (meaning that the subject has not yet been processed during the current search), the  
26 DataCounter is incremented (counting the number of subjects processed) and its value is  
27 placed in TransferVector[S] and pointer P. If the value of TransferVector[S] is non-zero,  
28 (meaning that MatchData already exists for the subject S) the value of TransferVector[S] is  
29 placed in pointer P. Subsequent operations are performed on MatchData[P][F].

1 If the bit representing minutia M is not set in the MinutiaBitMap of MatchData[P][F], it is set to  
2 indicate that minutia M has participated in the match, and AccumulateHough is invoked to  
3 calculate and accumulate Hough Transform data. The VisitCounter in MatchData is incremented  
4 to count the total number of visits to this Subject/Finger.

5 The AccumulateHough subprogram flow diagram is shown in Figure 19. The base angle and  
6 Cartesian coordinates for the Subject S, Finger F, Minutia M are obtained from the repository.  
7 The difference in base angles between the repository and search minutiae is calculated, giving  
8 DeltaBaseAngle. The repository Cartesian coordinates are rotated through DeltaBaseAngle  
9 (using standard trigonometric rotation techniques). The difference between the search minutia's  
10 coordinates and the rotated repository minutia coordinates is calculated giving DeltaX and  
11 DeltaY. DeltaX and DeltaY are quantized to the range 0..7 and used to increment the Hough  
12 accumulator HoughAcc [DeltaX][DeltaY].

13 The EvaluateMatch subprogram flow diagram is shown in Figures 20 and 21. The MatchData  
14 for each finger of each candidate subject is analyzed to create a raw score using the following  
15 equation:  $RawScore = (VisitCount - MinutiaCount) * (\max(HoughAcc[0..7][0..7]))$

16 The VisitCount variable effectively counts the number of minutia matches summed over all of  
17 the participating neighbor combinations. The MinutiaCount variable effectively counts the  
18 number of individual minutiae matches regardless of the source neighbor combination.

19 There are other possible means of calculating the raw score. The most promising of these are  
20 similar to the above equation but which replace the maximum HoughAcc value with the  
21 maximum of the sum of a cross pattern or a block pattern in the Hough Accumulator as in:

22	0 1 0	1 1 1
23	1 1 1	1 1 1
24	0 1 0	1 1 1

25 where we sum the neighboring cells which are indicated by 1's in the two pattern masks.

26 The raw score for each finger of each candidate subject is then normalized using statistical  
27 techniques appropriate for the observed exponential distribution of the raw scores. The  
28 standard deviation of the entire set of scores for the finger is calculated. The each raw score is

1 then divided by the standard deviation to produce a normalized exponential score for the  
2 Subject/Finger. The normalized exponential scores for each finger of each Candidate Mate are  
3 combined to produce a multi-finger score by summation. The Candidate Mates are scanned  
4 from first to last. If the multi-finger score for a particular Candidate Mate exceeds a pre-  
5 determined threshold, that subject is appended to the result list. The result list is then sorted in  
6 descending multi-finger-score order. The value of the threshold is a function of the operating  
7 point which produces the Reliability and Selectivity desired from the search process.

8 The output of the subprogram is a list of Candidate Mate subjects ordered from most likely  
9 (highest score) through least likely (lowest score).

10 Although the present invention has been described in terms of specific exemplary embodiments,  
11 it will be appreciated that various modifications and alterations might be made by those skilled in  
12 the art without departing from the spirit and scope of the invention as specified in the following  
13 claims.  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201